Will the Software Engineering Body of Knowledge (SWEBOK) replace to the Capability Maturity Model (CMM)???

The Software Engineering Body of Knowledge (SWEBOK) is an initiative started by the Institute of Electrical and Electronics Engineers (IEEE) to develop core set of tools that software professionals can use to do their job. Even with the number of software engineers/programmers exploding worldwide and the widespread presence of software in our society, software engineering has not reached the status of a legitimate engineering discipline. Since 1993, the IEEE Computer Society and the Association for Computing Machinery (ACM) have been actively promoting software engineering as a profession, mainly through their participation in the IEEE Computer Society and ACM Software Engineering Coordinating Committee.

The SWEBOK is still in the early phases of development and is under review. As of this writing, the 0.7 Stone Man version is available to reviewers for comment. This review will take place until late May. The next phase is the Iron Man phase and will last thru 2003. A formal version of this new IEEE standard will not be released until all reviews are performed and a general consensus is gathered.

The SWEBOK is composed of the following 10 topic areas:

- *Software Requirements*
- *Software Design*
- *Software Construction*
- *Software Testing*
- *Software Maintenance*
- *Software Configuration Management*

- *Software Engineering Management*
- *Software Engineering Process*
- *Software Engineering Tools and Methods*
- *Software Quality*

These topic areas seem to cover most of the CMM and more. I would like to briefly talk to each of these areas. The following brief descriptions come from the Stone Man 0.7 version of the SWEBOK.

The *Software Requirements* knowledge area is concerned with the acquisition, analysis, specification and management of software requirements.

The *Software Design* knowledge area refers to the transformation of the software requirements – typically stated in terms relevant to the problem domain – into a description explaining how to solve the software-related aspects of the problem. It describes how the system is decomposed and organized into components, and it describes the interfaces between these components. Design also refines the description of these components into a level of detail suitable for allowing their construction.

Software Construction is a fundamental act of software engineering; programmers must construct working meaningful software through coding, self-validation, and self-testing.

Software Testing consists of dynamically verifying a program's behavior on a finite set of test cases – suitably selected from the usually infinite domain of executions – against the specified expected behavior.

Software maintenance is defined as the totality of activities required providing cost-effective support to a software system whether they are performed pre-delivery or post-delivery.

We can define a system as a collection of components organized to accomplish a specific function and/or set of functions. A system's configuration is the function of physical characteristics of hardware, firmware, software, or a combination thereof as set forth in technical documentation and achieved in a product. *Configuration Management* is the discipline of identifying the configuration at distinct points in time to systematically control its changes and to maintain its integrity and traceability throughout the system life cycle.

Software Engineering Management addresses the management of software development projects and the measurement and modeling of such projects.

Software Engineering Process covers the definition, implementation, measurement, management, change, and improvement of software processes.

Software Engineering Tools and Methods knowledge area covers two topics that cut across the other knowledge areas: software tools and development methods. Software tools are the computer-based tools intended to assist the software engineering process. Tools are often designed to support particular methods, reducing the administrative load associated with applying the method manually. Like methods, they are intended to make development more systematic, and they vary in scope from supporting individual tasks to encompassing the complete life cycle.

The *Software Quality Assurance* process provides assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans. The software verification and validation process determines whether products of a given development or maintenance activity conform to the requirements of that activity and

those imposed by previous activities, and whether the final software product satisfies its intended use and user needs.

The CMM as it refers to software development, is broken down into 5 levels and 18 specific areas, from Level 2-Requirements Management to Level 5-Process Change Management. Based on the brief descriptions given for the SWBOK, I think we can make some correlations.

Software requirements relates to CMM Level 2 KPA of Requirements Management in the respect that they both come first in the process and they both consider proper requirements gathering to be critical to the success of a development effort. The SEI posts statistics relating to the success of software projects and the reasons for the failures. The majority of the failures relate to the lack of requirements gather and documented or if requirements are sought, they are never referred to after they are gathered. If requirements are not used after they are gathered, why take the time and effort to create the documentation. Either way, you don't know where you are going and will probably wind up with something the customer didn't really want or need. CMM Requirements Management goals 1 and 2 talk to the documenting and reviewing of requirements as the first phase of problem solving, you look for issues with the requirements before you go to design. Are the requirements traceable? Are they a need or a want? How do they relate to the real customer? Are the requirements coming from the customer (who is paying the bill) or the real user (the person who will have to use the system)? Whose requirements are more important?

The SWEBOK's *Software Design, Software Construction, and Software Testing* seem to have a loose relation to the Level 3 KPA Software Product Engineering (SPE). SPE goal 1 talks to the whole lifecycle from requirements to testing (basic waterfall), but not a lot of guidance is given

in any one area. Design is the second level of most software development life cycles, Construction relates to the code phase and of course testing to software test.

Software Configuration Management relates to the CMM Level 2 KPA of Software Configuration Management. The KPA talks to the control of the product, whether it is software, documentation or an interim component. SWEBOK version adds the references to hardware and firmware.

Software Engineering Management seems to relate to CMM Level 2 KPA of Software Project tracking and Oversight (PTO). Software Engineering Management consists of both the measurement/metrics and management process area. Metrics is a subject all software professionals should be familiar with. Management as defined by the SWEBOK relates to the activities that are undertaken in order to ensure that the software development process is performed in a manner consistent with the organization's policies, goals, and requirements (these issues seem to be shared by the CMM KPAs Organization Process Focus, Organization Process Definition, and Integrated Software Management). The CMM PTO talks to tracking actual results (goal 1), managing corrective actions (goal 2) and the changed agreed to between the affected groups (goal 3).

Software Engineering Process knowledge area relates to the CMM Level 3 KPA of Organizational Process Definition (OPD) and Level 5 KPA Process Change Management (PCM). Interestingly enough, most of the material adopted by the IEEE to cover this comes from the CMMI effort currently by a joint team of Government, academia and contractors to integrate the different CMM area (SW-CMM and SE-CMM; unfortunately not the SA-CMM) and reduce the redundancy that currently exists. IEEE also refers to another area Watts

Humphrey has developed, Team Software Process, in the generation of the materials for Software Engineering Process. With that in mind, I feel current and future software professionals will be able to migrate from one to the other with ease. OPD involves the collecting and maintaining of organizational standard software processes, documenting them into a resource library (goal 1) and providing the project-histories to the organization for future projects (goal 2). PCM involves software process evolution as a process; *Change is required. There is a process of change, just as there is a process of manufacturing, or for growing wheat. How to change is the problem.* (Dr. W. Edwards Deming, in the Forward to *The Team Handbook*, Madison, Wisconsin: Joiner Associates, 1992). Process Change Management involves planning change (goal 1), and team involvement in the process improvement process (goal2). SWEBOK Software Engineering Process breaks this process down into two activities: 1. The technical and managerial activities performed during development, maintenance, acquisition, and program retirement, and 2. The activities related to the definition, implementation, measurement, management, change and improvement of the software processes.

Software Engineering Tools and Methods knowledge area of the SWEBOK addresses the software development environments (computer-based tools intended to facilitate the software development process). These CASE tools are addressed to provide a methodology and administrative assist to ensure proper documentation is performed during the software development process. For some reason, documentation seem to be the least favorite part of t the development process, it helps having a system to self generate some of the very-necessary documentation. The knowledge area talks to documentation for all five phases of software development and the audits and reviews necessary for proper controls of that development process. The CMM doesn't talk specifically to CASE tools for software development, but it

talks to process, audits, reviews and inspections. Some of the areas are Level 2 KPA Software Quality Assurance for the control actions for project tracking, and the Level 4 KPA of Software Quality Management for the understanding of quality in the delivered products. All of these 'Quality' type activities require a plan and the monitoring of the software process to ensure proper controls are maintained.

Software Maintenance knowledge area of the SWEBOK deals with a very misunderstood part of the software lifecycle. Most managers do not plan on the costs or level of effort maintenance requires. For most projects, maintenance can take up 60 to 70% of the total costs of a software system. A common myth of software development is that when coding is finished and the software is released, the effort is finished. There is very little formal documentation or guidance on what it takes to properly maintain a software system. Most of the maintenance comes from a few organizational 'heroes' or from a contracted effort, with mixed results. The CMM refers to software maintenance in a few places, Level 2 KPA Software Configuration Management, Level 3 KPA Software Product Engineering and others. I fail to find a specific area in the CMM that refers directly to maintenance, except as a sub-set of another activity.

Finally we look at the SWEBOK knowledge area of *Software Quality*. The quality of the delivered product (artifact) requires a process to ensure the user and process requirements are met. There are so many references to quality in current literature today; the SWEBOK provides a distillation of the sources. It contains a list of the basic readings and additional references to some of the other noted writing. The CMM Level 2 KPA of Software Quality Assurance (SQA) and the Level 4 KPA of Software Quality Management (SQM) deal with the same area. SQA deals with the independent check for a 'management review' of the product and process. The

SQM portion of the CMM deals with process sharing and development of the best SQA tools and process for the organization.

The SWEBOK is a guide to the boundaries of software engineering designed to assist in generating a consensus among software professionals worldwide. As an IEEE standard, it will carry an inordinate amount of influence in the software world; therefore we need to ensure it get the attention it deserves while it is in development.

Will the SWEBOK replace the CMM? I feel the SWEBOK is taking much of its foundation from the CMM. I feel the SWEBOK covers a greater area than the CMM, but the IEEE cannot afford to miss any of the areas covered by Carnegie Mellon. The SWEBOK is, by design, not complete. Just like other engineering practices, there are some things an engineer is required to bring to the table. For example, software systems might be built in a specific language using functional modules (macros) in another language. You may write a business application in object COBOL and embed C++ modules to handle specific scientific calculations not available within the COBOL library suite. The techniques for integrating and configuring these type issues are skills and knowledge a software engineer needs to know and are not found in the SWEBOK. The SWEBOK is meant to be a source or reference guide covering the widest legitimate areas of software engineering without providing everything about any one area. There are additional references for supporting information to assist in any are where the guide is found wanting.

Very few people have a good handle on the entire CMM, even fewer are even aware of the SWEBOK. Only time will tell which is better to use. Maybe both are required just to support

the training of software professionals to ensure we attain true engineering professional status. The SWEBOK and its tie to the IEEE, will take us a long way in that direction.

References:

1.http://www.swebok.org/documents/stoneman06/Knowledge_Area_Description_for_Software_Requirements_Engineering(version_0_6).PDF

2. http://computing.db.erau.edu/SEERS/x.html

3. A guide to the CMM, Kenneth M. Dymond, Process Transition International Inc., 1998

4. Managing the Software Process, Watts S. Humphrey, Addison-Wesley, 1989

5. Software Engineering A Practitioner's Approach, Roger S. Pressman, McGraw-Hill, Inc., 1992